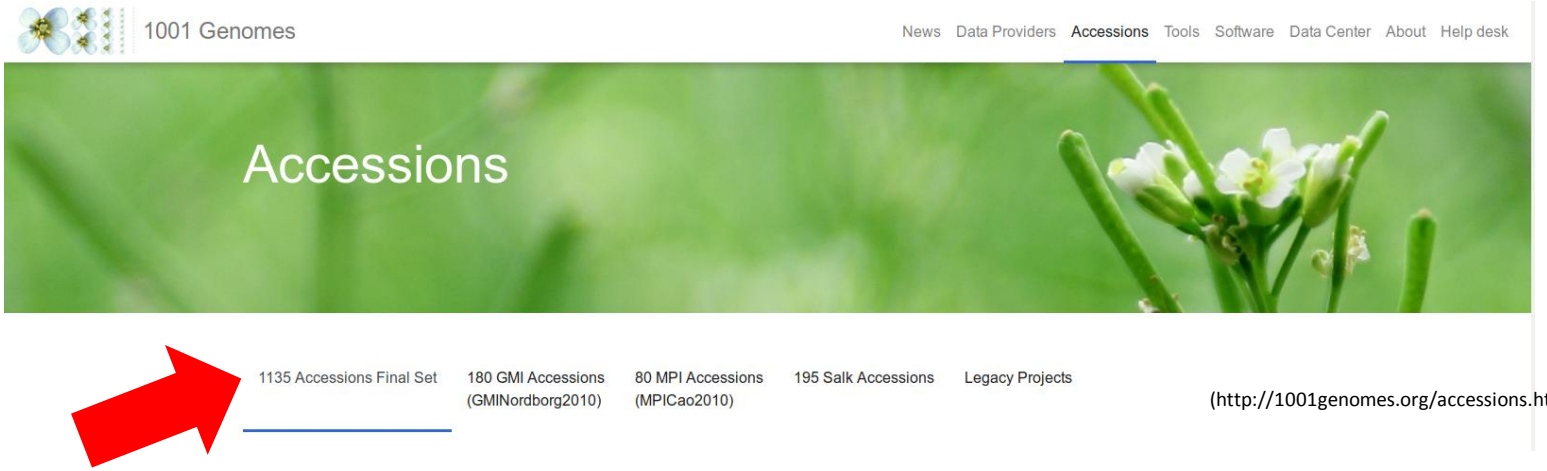
Applied Genome Research

Variant analysis

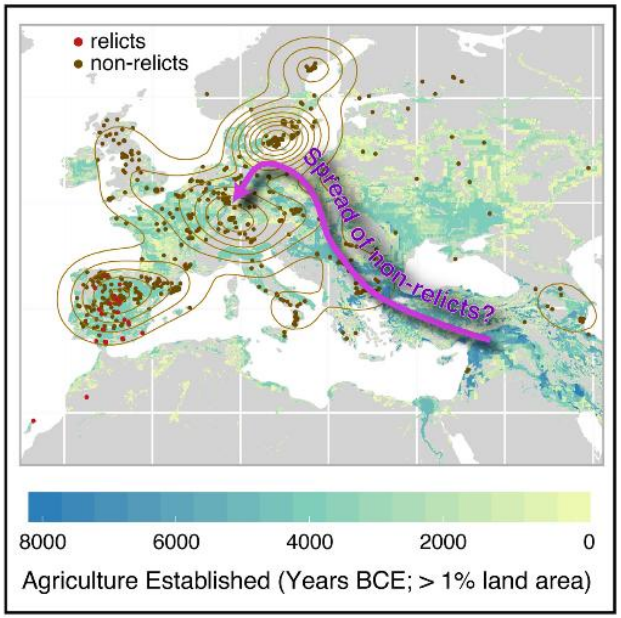
205048 & 205049

Boas Pucker

Re-sequencing



(http://1001genomes.org/accessions.html)



(https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4949382/)

Re-sequencing

- One high quality reference sequence per species is/was produced
- New strains are only analyzed by read mapping against the reference sequence
- SNPs and small InDels can be detected
- Detection of large rearrangements and InDels is complicated
- Identification of new genes requires *de novo* assembly

Mapping

- Mapping = alignment of sequencing read(s) with reference sequence
- BLAST is to slow to compute alignments for billions of reads
- Burrows-Wheeler Alignment tool (BWA) is well suited for NGS read mapping

BWA - usage

$bwa index <reference_file> … preparing reference for mapping

$bwa mem \ … calls special version of this mapper for long reads

–M \ … low quality hits are flagged as secondary hits

–t <INTEGER> \ … number of threads to use

<reference_file> \ … reference file as specified before

<read1> \ … file with forward reads

<read2> \ … file with reverse reads (optional)

| gzip -3 > <output_file> …. addition to produce compressed results

BWA – follow up

Conversion of BWA result (SAM) into BAM file:

$samtools view –bS –F 4 \

<SAM_file> > <BAM_file>

-b = bam file is output

-S = SAM file is input

-F 4 = reads with flag ‘4’ (bad quality) are discarded

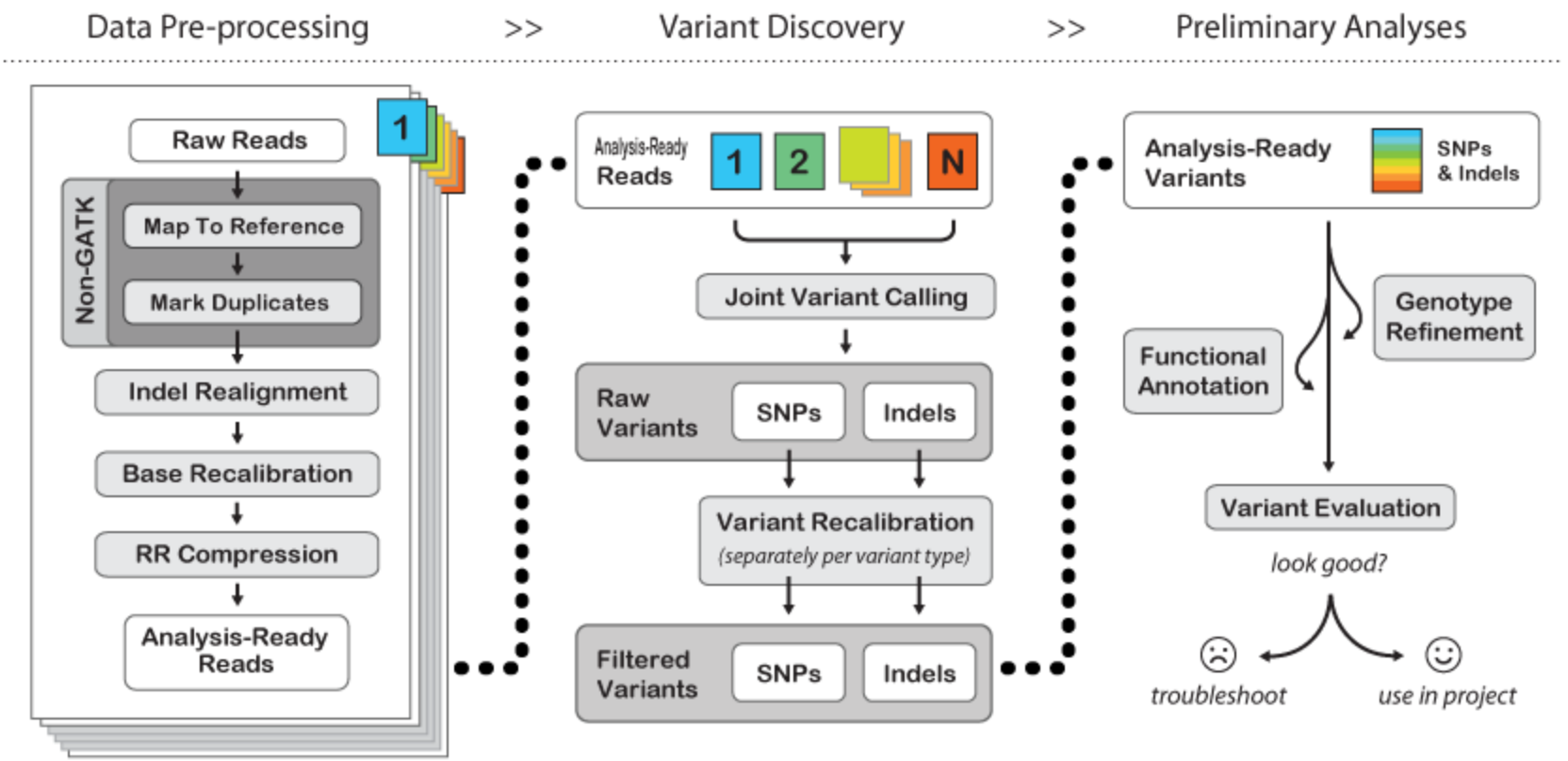
EXERCISE

- Use BWA MEM for Nd-1 sequencing read mapping against the provided Col-0 reference sequence and convert the SAM result file into BAM format!
- Which statistics about the mapping result are available?

Variant calling

- Detection of differences between reference sequence and reads
- Only small variants can be detected, because large variants prevent the reads from mapping

GATK - concept



(https://gatkforums.broadinstitute.org)

GATK – index reference file I

$ /vol/java-8/bin/java –Xmx8g –jar picard.jar \ … call tool

CreateSequenceDictionary \ … select operation to perform

R=<ref_file> \ … reference file

O=<dict_file> ….reference file + ‘.dict’ (without .fasta extension!)

GATK – index reference file II

$ samtools \ …. calls tool

faidx \ … selects operation

<ref_file> … reference file to index

(previously: R=<ref_file> … reference file to index)

GATK – sorting BAM file

$ samtools sort \ … calling tool and selecting method

-@ 4 \ …. specifying number of CPUs to use

-m 4G \ … setting amount of RAM to use

-o <output_file> \ … specifying output file name

<input_file> …. input file is passed as last element

GATK – indexing BAM file

$ /vol/java-8/bin/java –Xmx8g –jar picard.jar \ … calling tool

BuildBamIndex \ … selecting one method

I=<bam_file> … providing input file

GATK – add readgroup

$ /vol/java-8/bin/java –Xmx8g –jar picard.jar \ … calling tool

AddOrReplaceReadGroups \ …. selecting method

I=<input_file> \ … providing input file

O=<output_file> \ …. setting output file

RGID=1 RGLB=lib1 RGPL=illumina RGPU=unit1 RGSM=20 … some values to use as new read group information

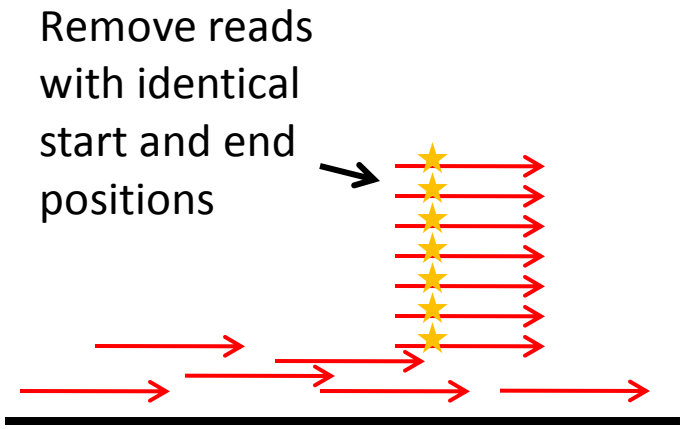
ReadGroups:
RGID = ID
PL = read group platform lane
LB = read group library
PU = read group platform unit (e.g. barcode)
SM = read group sample name

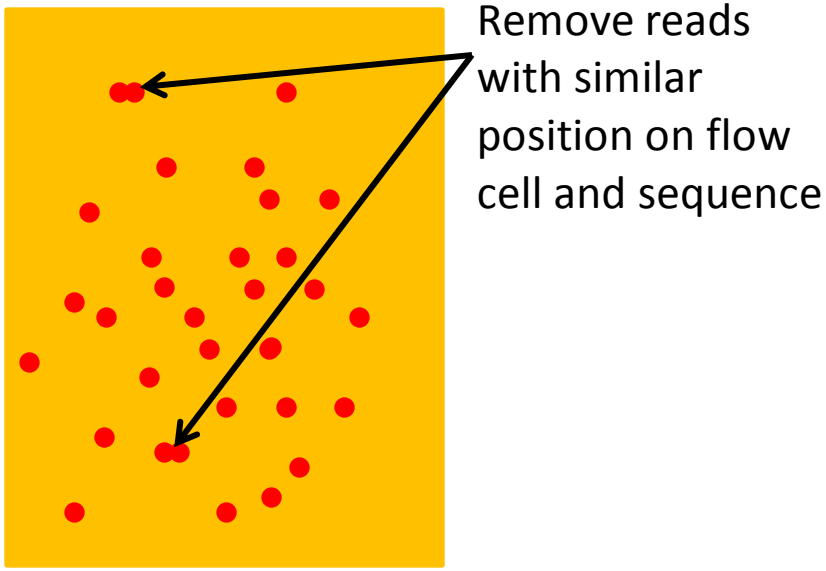
GATK – indexing BAM file

```
$ /vol/java-8/bin/java –Xmx8g –jar picard.jar \ … calling tool
 BuildBamIndex \ … selecting one method
 I=<bam_file> \ … providing input file
```

PCR duplicates

- Library preparation involves PCR
- Amplification of erroneous molecules possible





GATK – mark duplicates

$ /vol/java-8/bin/java –Xmx8G –jar picard.jar \ … calling tool

MarkDuplicates \

INPUT=<bam_file> \

OUTPUT=<marked_duplicate_file> \

METRICS_FILE=<metrics_file> \

OPTICAL_DUPLICATE_PIXEL_DISTANCE=2500 \

CREATE_INDEX=true TMP_DIR=/tmp

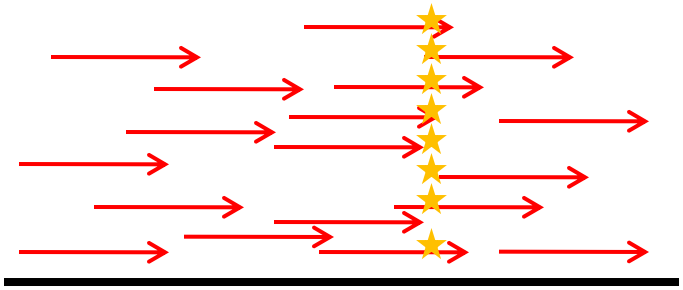
GATK – get realignment targets

$ /vol/java-8/bin/java –Xmx8g –jar GenomeAnalysisTK.jar \ … tool
-T RealignerTargetCreator \ … selecting operation
-R <reference_seq_file> \ … providing reference sequence
-I <bam_file> \ …. providing prepared BAM file
-U ALLOW_N_CIGAR_READS \ … option is need to prevent errors
-o <result_file> … setting output file (.list extension is needed!)

This step might not be working on compressed files!

InDel realignments

- Reads are mapped erroneous around InDel
- Local *de novo* assembly resolves situation



GATK – get realignments

$ /vol/java-8/bin/java –Xmx8g –jar GenomeAnalysisTK.jar \ … tool
-T IndelRealigner \ … specifying method
-R <reference_file> \ … providing reference sequence
-I <bam_file> \ … prepared BAM file
-targetIntervals <target_interval_file> \ … prepared target file
-U ALLOW_N_CIGAR_READS \ … needed to prevent errors
-o <output_file> … setting output file

GATK - HaplotypeCaller

```
$ /vol/java-8/bin/java –Xmx8g –jar GenomeAnalysisTK.jar \ … tool
 -T HaplotypeCaller \
 -R <reference_file> \
 -I <bam_file> \
 --genotyping_mode DISCOVERY \
-stand_emit_conf 10 \
-stand_call_conf 30 \
-U ALLOW_N_CIGAR_READS \
-o <output_file> #VCF
```

GATK – extract SNPs

$ /vol/java-8/bin/java –Xmx8g –jar GenomeAnalysisTK.jar \ … tool

-T SelectVariants \ … selecting method

-R <ref_file> \ … providing reference sequence

-V <raw_vcf_file> \ … providing raw variant file

-selectType SNP \ … select only SNPs

-o <raw_snp_vcf_file> …. set SNP vcf file

GATK – filter SNPs

```
$ /vol/java-8/bin/java –Xmx8g –jar GenomeAnalysisTK.jar \ … tool
 -T VariantFiltration \ … selecting method
 -R <reference_file> \ … providing reference sequence file
 -V <snps_only_vcf_file> \ … providing variant data
 --filterExpression “QD < 2.0” --filterName “QD_filter” \
 --filterExpression “FS > 60.0” --filterName “FS_filter” \
 --filterExpression “MQ < 40.0” --filterName “MQ_filter” \
 -o <clean_snp_vcf_file> … final output file
```

GATK – extract InDels

$ /vol/java-8/bin/java –Xmx8g –jar GenomeAnalysisTK.jar \ … tool

-T SelectVariants \ … selecting method

-R <ref_file> \ … providing reference sequence

-V <raw_vcf_file> \ … providing raw variant file

-selectType INDEL \ … select only InDels

-o <raw_indel_vcf_file> …. set InDel vcf file

GATK – filter InDels

```
$ /vol/java-8/bin/java –Xmx8g –jar GenomeAnalysisTK.jar \ … tool
 -T VariantFiltration \ … selecting method
 -R <reference_file> \ … providing reference sequence file
 -V <indels_only_vcf_file> \ … providing variant data
 --filterExpression “QD < 2.0” --filterName “QD_filter” \
 --filterExpression “FS > 200.0” --filterName “FS_filter” \
 -o <clean_indel_vcf_file> … final output file
```

EXERCISE

- Run GATK on BWA mapping results!
- Calculate frequency of SNPs and InDels for the resequenced interval!
- Compare the SNP and InDel frequency to values from the literature!

Variant annotation

- Impact of mutation depends on type and position
- One mutation could effect multiple genes in different ways
- Most high impact mutations are located in coding sequences
- Functional annotation via impact on encoded gene product

SnpEff

- Uses annotation from GFF file
- Annotates each variant in VCF file by creating a new commented VCF file
- Very fast, but effect association is limited to one variant at a time

SnpEff - usage

```
$ java -Xmx4g -jar snpEff.jar \
  -v athaliana130 \
 <input_vcf_file> \
 > <output_vcf_file>
```

... calling tool

.... database to use for annotation

... vcf file contains variants to annotate

... annotated variants

SnpEff - results

Contents

More information:
http://snpeff.sourceforge.net/SnpEff_manual.html

SnpEff: Variant analysis

ANN field (VCF output files)

Functional annotations information is added to the INFO field using an `ANN` tag. The annotation 'ANN' field looks like this (the full annotataion standard specification can be found here).

```
ANN=T|missense_variant|MODERATE|CCT8L2|ENSG00000198445|transcript|ENST00000359963|protein_coding|1/1|c.1406(
```

A variant can have (and ususally has) more than one annotaion. Multiple annotations are separated by commas. In the previous example there were two annotations corresponding to different genes (CCT8L2 and FABP5P11).

Each annotation consists of multiple sub-fields separated by the pipe character "|" (fields 15 and 16 are empty in this example):

```
Annotation      : T|missense_variant|MODERATE|CCT8L2|ENSG00000198445|transcript|ENST00000359963|protein_cod
SubField number : 1|       2        |    3   |  4   |       5       |    6     |      7        |      8
```

Here is a description of the meaning of each sub-field

1. **Allele (or ALT):** In case of multiple ALT fields, this helps to identify which ALT we are referring to. E.g.:

```
#CHROM  POS     ID  REF  ALT    QUAL  FILTER  INFO
chr1    123456  .   C    A      .     .       ANN=A|...
chr1    234567  .   A    G,T    .     .       ANN=G|... , T|...
```

In case of cancer sample, when comparing somatic versus germline using a non-standard reference (e.g. one of the ALTs is the reference) the format should be ALT-REFERENCE. E.g.:

```
#CHROM  POS     ID  REF  ALT  QUAL  FILTER  INFO
chr1    123456  .   A    C,G  .     .       ANN=G-C|...
```

Compound variants: two or more variants affecting the annotations (e.g. two consecutive SNPs conforming a MNP, two consecutive frame_shift variants that "recover" the frame). In this case, the Allele field should include a reference to the other variant/s included in the annotation:

```
#CHROM  POS     ID  REF  ALT  QUAL  FILTER  INFO
chr1    123456  .   A    T    .     .       ANN=T|...
chr1    123457  .   C    G    .     .       ANN=C-chr1:123456_A>T|...
```

2. **Annotation (a.k.a. effect):** Annotated using Sequence Ontology terms. Multiple effects can be concatenated using '&'.

```
#CHROM  POS     ID  REF  ALT  QUAL  FILTER  INFO
chr1    123456  .   C    A    .     .       ANN=A|intron_variant&nc_transcript_variant|...
```

Genome wide variation distribution

Variants rate details

Chromosome	Length	Variants	Variants rate
1	30,427,671	128,436	236
2	19,698,289	86,309	228
3	23,459,830	94,376	248
4	18,585,056	83,031	223
5	26,975,502	94,335	285
Total	**119,146,348**	**486,515**	**244**

Number variantss by type

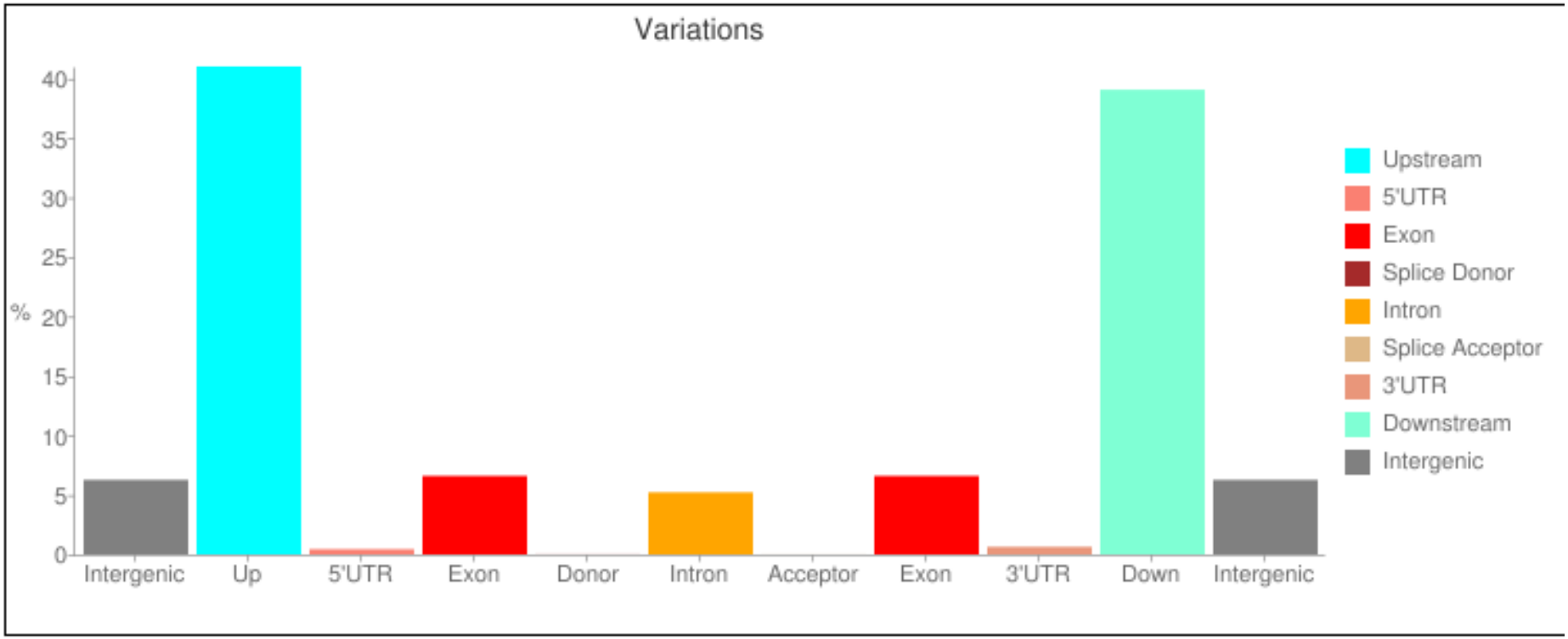
Type	Total
SNP	410,093
MNP	0
INS	36,949
DEL	39,473
MIXED	0
INTERVAL	0
Total	**486,515**

Number of effects by impact

Type (alphabetical order)	Count	Percent
HIGH	5,237	0.135%
LOW	133,414	3.448%
MODERATE	92,802	2.399%
MODIFIER	3,637,473	94.018%

(source: Pucker et al., 2016)

Variations over gene structure



(source: Pucker et al., 2016)

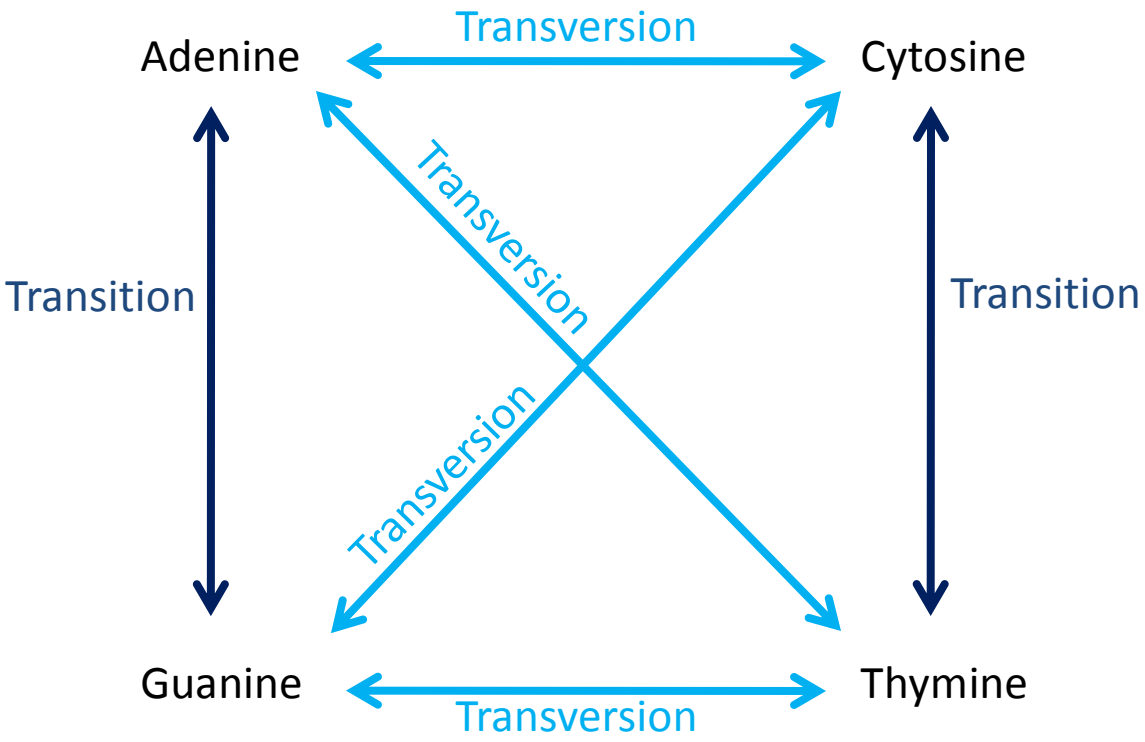
Base change frequencies

Base changes (SNPs)

	A	C	G	T
A	0	21,993	57,833	31,019
C	21,458	0	15,748	56,960
G	57,362	15,670	0	21,674
T	31,094	57,538	21,744	0

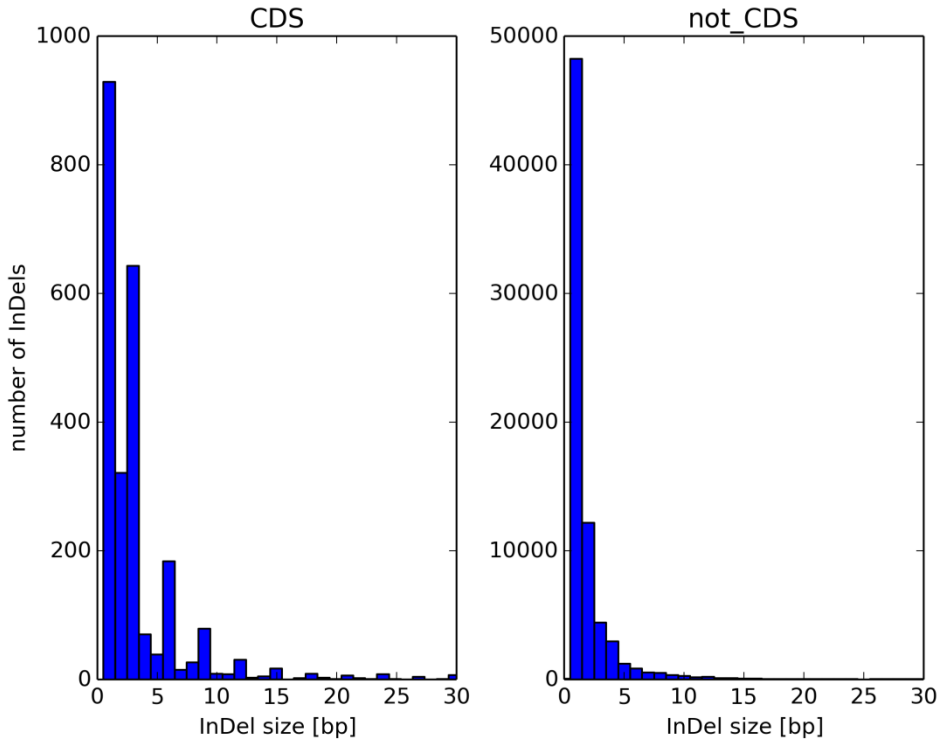
(source: Pucker et al., 2016)

Transition and transversion



InDel length distribution

- Small InDels are more frequent than larger events
- InDel length of 3 or multiple times 3 is neutral in CDS



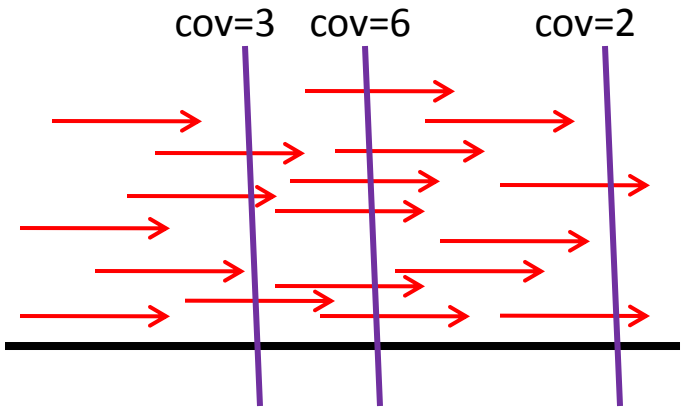
(source: Pucker et al., 2016)

QUESTION

- Which mutation types are highly deleterious?
- How many critical mutations are there in the resequenced interval?
- How could we select the most important effect of each mutation?
- What are the functions of the mutated genes? Select some examples and investigate possible phenotypes.
- Why are the bases not changing in a completely random pattern?

Coverage

Number of reads that span a specific position in the genome



Analyze coverage

- Calculate coverage based on BAM file:

```
python construct_coverage_file.py \
--in final_mapping_file.bam \
--out coverage_file.cov \
--bam_is_sorted
```

- Visualize coverage of a region:

```
python cov_figure.py \
--cov coverage_file.cov \
--chr Chr5 \
--start 100 \
--end 500
```

- Scripts are available at: https://github.com/bpucker/script_collection

EXERCISE

- Generate coverage file based on final mapping file (Nd-1 vs. Col-0)!
- Visualize region in Col-0 which should be covered by Nd-1 reads!
  - (Chr5, 5Mbp-8Mbp)